2.1 CONFIGURATION

Platform configuration refers to the physical attributes that have some effect on the movement, susceptibility, vulnerability, and system-based capabilities of the platform. In Suppressor, these physical attributes are implemented as the user-defined physical and logical relationships between the components and capabilities of a player. The user implements the appropriate configuration of each simulation entity by creating a player to represent the entity and then specifying the elements, systems, and expendables that will implement the real-world platform's attributes and capabilities. The user can structure the player so that all of its components have the same location and, if movement is possible, the same movement path. On the other hand, the player can also be defined with elements at several locations, each with separate movement capability. Similarly, the user can specify that elements at one location be treated as a single entity or as multiple entities with respect to susceptibility and vulnerability. In addition, the user may define the systems (movers, sensors, talkers, thinkers, weapons, or disruptors) and their capabilities and expendables in each element.

2.1.1 Functional Element Design Requirements

The design requirements for implementing platform configuration attributes are:

a. Provide the user with sufficient flexibility in defining the player structure so that its configuration produces the user-desired effects on the attributes and capabilities of the simulated platform. This includes a requirement to allow each player to have components at one or more locations, with one or more separately susceptible and/or vulnerable components at each location, and zero or more system capabilities for each separate component at each location. In addition, each system may have zero or more expendables.

b. The player structure must be susceptible to detection and damage from other players. Components under the location are the lowest level that can be sensed and killed. A location with zero remaining compoennts is considered to be killed. A player with zero remaining locations is considered to be dead. As each component is killed the capabilities of its systems are lost.

c. Provide the following generic types of systems: movers, thinkers, talkers, sensors, weapons, and disruptors. Suppressor will provide the capability for the user to specify the capabilities of each generic system type.

2.1.2 Functional Element Design Approach

In Suppressor, characteristics of a player are defined in the Type Data Base (TDB) with a PLAYER-STRUCTURE data item. This data item is basically a "table of contents" for the player type. It defines the specific hierarchy of locations and equipment owned by that player type. In the Scenario Data Base (SDB), the scenario developer populates the scenario with instances of the player types defined in the TDB PLAYER-STRUCTUREs.

The components which are used to define a PLAYER-STRUCTURE are as follows:

PLAYER - The fundamental unit in a Suppressor scenario. Players interact with other players using one or more of the generic functions: sensing, shooting, disrupting, talking,

moving, noticing, digesting, reacting. In the scenario laydown, players form command chains and command chains form sides.

LOCATION - Each player contains one or more locations. The main attribute of a location is a geographical position in the scenario space. The actual position, given in either Cartesian or spherical coordinates, is specified in the SDB definition for an instance of the player type. Typically, a location is a part of a player which performs a certain activity. For example, a Surface-to-Air Missile (SAM) player might have a location devoted to a battery commander, other locations possessing the radars, and other locations for the missile launchers.

ELEMENT - Each location contains one or more elements. The element is the targetable entity within a player. Signatures (e.g. radar cross section, optical cross section, and infrared radiance) are defined on a per-element basis. Finally, elements are placeholders for systems. For example, assume that an airplane has two elements at its only location. One element might contain sensor systems and the other the weapon systems. Since an incoming ordnance is directed at a single element, it would be possible for the weapon element to be killed and the sensing element to remain living, thus degrading the overall capability of the aircraft.

One final word on elements: Elements are considered to be collocated with their parent locations for positional purposes. Therefore, in the preceding example, although the elements might appear to the observer to be in different "places" on the airplane, Suppressor would treat them as being at the same geographical position (that given to the parent location) when performing range calculations, etc.

SYSTEM - Each element contains zero or more systems. There are eight system types defined in Suppressor:

a. Communications Receiver
b. Communications Transmitter
c. Disruptor
d. Mover
e. Sensor Receiver
f. Sensor Transmitter
g. Thinker
h. Weapon

Suppressor provides sets of system type-dependent input items which are used by the scenario developer. Implementation details of the system types are given in the functional element descriptions below.

RESOURCE - Some system types can have associated resources, also known as expendables. Mover systems can expend fuel. Weapon systems expend ordnance at the target. Disruptor systems (those which are defined implicitly) can expend a generic CM-EXPENDABLE which affects target tracking capability. Finally, thinker systems can dynamically create a new player in response to a launch (i.e. scramble) message from a commander.

Unlike other models, Suppressor makes no assumptions as to how a particular player type is structured. The scenario developer has total flexibility in determining the number of locations a player has, as well as the configuration of the elements and systems it owns. This flexibility requires that the user decide whether to model an entity as a single player type with multiple locations or as multiple player types, each having fewer locations. One of the main considerations in this decision is whether there is a need to model explicit communications traffic between the locations. If there is such a need, then multiple player types are necessary because messages are passed only between separate players, not between locations within a single player.

2.1.3 Functional Element Software Design

The Suppressor player structures are created in module KNDSTR. There is a separate player structure data block for each player type in the TDB. The player structure data block is called the central directory (data block 1) and contains pointers to player location, element, system, expendables, and tactics data as listed in Table 2.1-1. The central directory is pointed to from the interaction key (data block 39) which can be accessed in various ways including through the model treetop (data block 100).

TABLE 2.1-1. Data Pointers defined in Player Structure Data Block.

Modules	Data Block (number)
LCMDL	ptr to command, control list (37)
LNKEY	ptr to interaction key (39)
LJAMR	ptr to jammer system list (92)
LTACT	ptr to tactics, doctrine (96)
LWPNS	ptr to weapon system list (51)
LFOES	ptr to sensor perception list (17)
LSUBS	ptr to subordinate perceptions list (10)
LORDS	ptr to orders/assignments (18)
LCLST	ptr to cluster list (2)
LPAIL	ptr to player acton item list (33)
LMNTL	ptr to mental processing (22)
LSENR	ptr to sensor heading list (46)
LCOMM	ptr to communication status (69)
LFREN	ptr to friendly player list (10)
-----	reserved for HARM flyout
-----	reserved for HARM flyout
-----	reserved for HARM flyout

The design for processing a PLAYER-STRUCTURE in a TDB is implemented in module KNDSTR:

```
*when player structure declaration (521100):
   *meld player structure into list;
   *when replacement mode:
      *when data had been defined:
         *recycle old structure;
      *end of test for data defined.
   *otherwise, data should be new:
      *set error flag if data present;
   *end of test for mode.
   *signal an error to user if necessary;
   *allocate dummy central directory;
   *when input name is not a player name, notify user;
```

```
*but, when structure data groups declared (522000):
   *meld data group header into list;
   *meld data group buffer into structure list;
   *when input is not a capability but should be, notify user;
   *when input is not a tactic but should be, notify user;
   *when input is not a susceptibility but should be, notify user;
*but, when structure location (523000):
   *add structure to player list;
*but, when structure element (524[0,1]00):
   *when element sentence (524000):
      *allocate element, add to list;
      *allocate dummy element, store pointer;
      *when input is not an element, notify user;
   *but, when criticality phrase (524100):
*but, when structure for system (525000):
   *allocate system, add to list;
   *allocate dummy data blocks for system;
   *store pointer to character string for name;
   *when input name does not match system type, notify user;
*but, when resource structure for systems (526x00):
   *allocate resource, add to list;
   *when traditional resource:
      *when resource is real-valued:
         *convert to internal units;
      *end of test for real value.
   *but, when external name:
      *store pointer to character string for name;
      *when input name is not an external name, notify user;
   *end of test for resource class.
*but, when structure linkages for systems (527000):
   *allocate linkage, add to list;
   *loop, while system ID's to find:
      *search locations for system;
      *signal an error if not found;
   *end of loop for system ID's.
*end of test for type of input.
```

2.1.4 Assumptions and Limitations

Player representations have no physical dimensions. They are modeled as point objects at the locations specified.

2.1.5 Known Problems or Anomalies

None.